Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 78 (2016) 683 – 690

Procedia Computer Science

International Conference on Information Security & Privacy (ICISP2015), 11-12 December 2015, Nagpur, INDIA

Tree Representation: Knowledge Discovery from Uncertain Data

K. Swarupa Rani*

School of Computer and Information Sciences, AI Lab, University of Hyderabad, Hyderabad, 500046, India

Abstract

Most of the real-world data is gathered locally, organized regionally leading to distributed environment. The analysis of such data will assist decision makers for promoting their business. Most of the data mining strategies are multi-pass techniques employed for mining and discovering the knowledge. Hence, the paper focuses on developing two pass data mining constructs to handle uncertain data. One of the simplest and feasible item-pattern representations is transactional tree. The developed methodology for constructing transactional trees is studied in detail and apt technological solutions is proposed. It is proposed to construct transactional tree from uncertain data which is usually associated with existential probabilities and to establish distributed computing environment for developing transactional trees. This paper aims at bringing out the knowledge valid for a given location through which it is expected to derive global knowledge.

Peer-review under responsibility of organizing committee of the ICISP2015

*Keywords:*Uncertain Data; Compact PUF Tree; Parallel Construction; Transactional Tree;

1. Introduction

Data Mining aims to discover knowledge from data. Uncertain data is prevailing in all these applications. Uncertainty can be caused by our limited perception or understanding of reality. Researchers contributed and paid more attention on mining frequent patterns from probabilistic database of uncertain data and developed algorithms to mine frequent patterns from transaction databases of uncertain data. The real world data is of two categories, namely precise data and uncertain data. In precise data, user is aware about the presence or absence of an item or an event in the transactional data. In uncertain data the presence or an absence of an item in a transaction is not certain and users confidence of membership is expressed using existential probabilities.

* Corresponding author. Tel.: 040-23134105.
*E-mail address:*swarupacs@uohyd.ernet.in

Peer-review under responsibility of organizing committee of the ICISP2015
doi:10.1016/j.procs.2016.02.117

Researchers Chui et al. [1] , Dai et al. [2], Leung et al. [3] worked on uncertain data. Discovery of Knowledge from uncertain data is essential in real-life situations and one need to develop efficient algorithms to meet the demands of mining uncertain data. Chui et al. [1] developed U-Apriori which relies on the candidate generate-and-test paradigm. Leung et al. [3] developed tree structure for mining uncertain data and enhanced the works in [4] with UF-tree and UF-growth algorithm. Handling and compiling uncertain data in single machine will be voluminous. Thus, distributed computing strategies are essential and further optimal representation strategies are needed. In this paper the distributed computing strategies are used to represent the transactional tree in distributed locations from which global transactional tree can be obtained. Based on these factors, this paper modified and extended the works of Leung et al. [8] in order to represent the transactional tree through Compact PUF-Tree in distributed environment and the same is presented in Section 4. This contribution further leads to extract frequent patterns in order to derive local and global knowledge by applying mining algorithms.

The remainder of this paper is as follows: Section 2 briefly discusses the parallel and distributed environments to construct the Compact **P**refix-capped **U**ncertain **F**requent pattern tree (Compact PUF-Tree) to meet the challenges associated with uncertain data. Section 3 deals with the construction of Compact PUF-Tree. Section 4 deals with distributed methodology of Compact PUF-Tree construction, Section 5 with experimentation and analysis. Section 6 with conclusions.

2. Related Works

Many existing algorithms mine frequent patterns from precise data. Researchers paid attention to handle uncertain data to mine frequent patterns from transactional tree. According to Chui et al. [1] U-Apriori algorithm is an Apriori based algorithm proposed to handle uncertain data and it requires number of scans. Leung et al. [4] proposed tree-based UF-growth algorithm in order to reduce the number of scans of uncertain DBs. UF-growth algorithm consists of two operations: (1) Tree construction (UF Tree) and (2) Mining frequent patterns from UF-Tree. UF-Tree suffers from few problems. Tree can be shared only if nodes on the path have same item with same existential probability in order to compute expected support of each pattern. To overcome this problem, Aggarwal et al. [6] proposed UFP-Growth algorithm in which nodes are grouped based on the same item and similar existential probability values. UFP-Growth algorithm extracts frequent and infrequent patterns.

To overcome these problems Leung et al. proposed Prefix-capped Uncertain Frequent pattern tree (PUF-Tree) [8] which is compact as FP-Tree [10]. In this paper PUF-Tree [8] mechanism is extended. The background information is given below in order to construct Compact PUF-Tree in distributed and parallel environment.

- Existential Probability: The presence of an uncertain item ‘x’ in a transaction ‘t’ is given by existential probability, P(x€t) € (0,1)
- Expected support: expSup(X) in DB is the sum of $P(X,t_j)$ over all ‘n’ transactions in DB.

$$\text{expSup}(X)=\sum_{j=1}^{n} P(X,t_j)$$

$$= \sum_{j=1}^{n} \left(\prod_{x \in X} P(x, t_j)\right)$$

- Prefixed Item cap $I^{Cap}(x_r, t_j)$: The item cap for an item x_r in $t_j = \{x_1, \ldots\ldots x_r, \ldots.. x_h\}$ is the product of $P(x_r, t_j)$ and the highest existential probability value ‘M’ of items from x_1 to x_{r-1} in t_j

$$I^{Cap}(x_r, t_j) = \begin{cases} P(x_r, t_j) * M \text{ if } h > 1, & \text{where } M = \max_{1 \le q \le r-1} P(x_q, t_j) \\ P(x_r, t_j) \quad \text{if } h=1 \end{cases}$$

3. Construction of PUF-Tree[8]

Consider the uncertain transactional database from Table 1 and let minsup=0.5. According to Table 1 the Itemset={a,b,c,d,e,f} where m=6. Uncertain DB={t_1, t_2, t_3, t_4} be the set of four transactions where each $t_j \subseteq$ Item. PUF-Tree is constructed using two scans of the DB. First, scans the DB for calculating the expSup of each item.

Second, each t_j is scanned and inserted into the tree.

Table 1.Transactional Uncertain DB.

TID	Transactions
t_1	{a:0.2,b:0.2,c:0.7,f:0.8}
t_2	{a:0.5,c:0.9,e:0.5}
t_3	{a:0.3,d:0.5,e:0.4,f:0.5}
t_4	{a:0.9,b:0.2,d:0.1,e:0.5}

The following are the expected support of each item in transactions of Table 1. Expected support of 1-itemset length pattern where X = {a}

expSup(a)=$\sum_{j=1}^{4} (\prod_{a \in X} P(a, t_j))$

= $P(a,t_1) + P(a,t_2) + P(a,t_3) + P(a,t_4)$

=0.2 + 0.5 + 0.3 + 0.9 =1.9

Similarly, calculating the expected support for remaining items we get expSup(b) = 0.4, expSup(c) = 1.6, expSup(d) = 0.6, expSup(e) = 1.4, expSup(f) = 1.3.

The following Table 2 gives the I-list and updated I-list which contains the expected support of 1-itemset

Table 2.I-List and Updated I-List

I-List	
Item	expSup
a	1.90
b	0.40
c	1.60
d	0.60
e	1.40
f	1.30

Updated I-List	
Item	expSup
a	1.90
c	1.60
e	1.40
f	1.30
d	0.60

The items are removed from I-List of Table 2 which is not satisfying the minsup. Hence, 'b' item is removed and remaining items are sorted in descending order. Based on Updated I-List, Table 1 should be changed accordingly. The following are the I^{Cap} values of updated transactions in Table 1

$I^{Cap}(a,t_1) = 0.2$

$I^{Cap}(c,t_1) = 0.7 * \max\{a:0.2\} = 0.14$

$I^{Cap}(f,t_1) = 0.8 * \max\{a:0.2, c:0.7\} = 0.56$

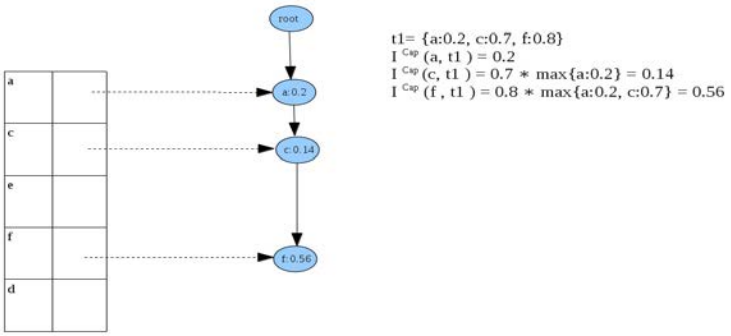
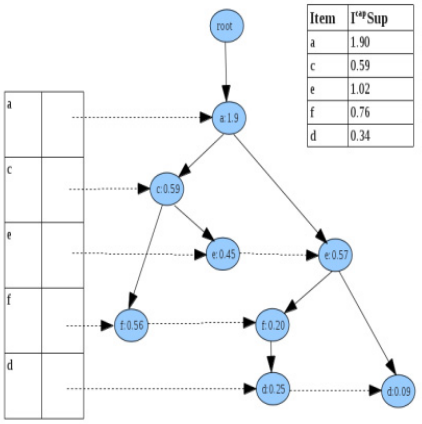


Fig. 1. Transaction $t_1$is inserted into the PUF-Tree.

Each node in PUF-Tree contains an item name and its I^{Cap} value. If an item of the transaction and nodes of the tree from the root are same then update the item cap value of the node. Else create a new node with corresponding I^{Cap} value. The following Fig.1 is the PUF-Tree containing transaction t_1. Similarly after inserting the remaining transactions such as t_2, t_3, t_4 the resultant PUF-Tree with I^{Cap} Support is shown below in Fig 2. Remove item 'd'

from PUF-Tree as I^{cap} Sup(d) < minsup. The following Fig. 3 is the compact PUF-Tree by pruning the item 'd'. This technique saves the mining time.



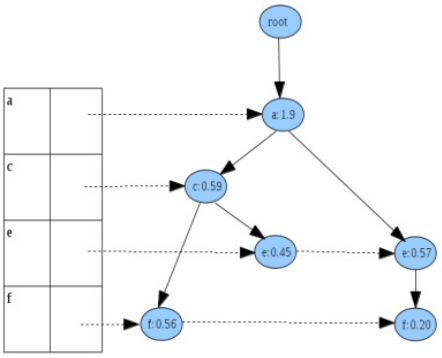


Fig. 2. PUF-Tree with I^{cap} support Fig. 3. Compact PUF-Tree.

4. Distribution and Parallel Construction of Compact PUF-Tree

The construction time of the PUF-Tree [8] to handle large transactions is huge and can be reduced effectively by adopting distributed computing strategies. There is a need to derive frequent patterns in different locations and consolidate them to achieve global patterns. In this scenario, one needs to adapt distributed computing paradigm. To overcome the existing limitations, PUF-Tree algorithm of Leung et al. [8] is modified and extended for constructing PUF-Tree parallelly and an algorithm is proposed to merge sub PUF-Trees in order to obtain Global Compact PUF-Tree (GPUF-Tree). The following are the steps involved in the proposed approach:

Step 1: Divide the dataset and parallely deploy to different nodes
Step 2: Each node parallely computes expSup of each item (i.e., I-List) and communicates to Central Server.
Step 3: Central Server compiles and accumulates (I-List) from different nodes and generates (G-List) and distributes to all the local nodes. (G-List is same as Updated I-List which is shown in Table 2).
Step 4: On receipt of (G-List) from Central Server, each local node builds Compact PUF-Tree parallely and communicates back to Central Server.
Step 5: Central Server merges, local PUF-Trees to obtain Global Compact PUF-Tree (GPUF-Tree).

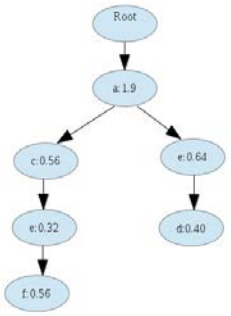
Table 3.Sample Database of various distributed locations

Location 1		
TID	Items	Order
t_1	a:0.2, b:0.2, c:0.7, f:0.8	a:0.2, c:0.7, f:0.8
t_2	a:0.5, c:0.9, e:0.5	a:0.5, c:0.9, e:0.5
t_3	a:0.3, d:0.5, e:0.4, f:0.5	a:0.3, e:0.4, f:0.5, d:0.5
t_4	a:0.9, b:0.2, d:0.1, e:0.5	a:0.9, e:0.5, d:0.1

Location 2		
TID	Items	Order
t_5	a:0.3, b:0.3, c:0.8	a:0.3, c:0.8
t_6	a:0.4, c:0.8, e:0.4, f:0.7	a:0.4, c:0.8, e:0.4, f:0.7
t_7	a:0.4, d:0.6, e:0.4	a:0.4, e:0.4, d:0.6
t_8	a:0.8, b:0.1, d:0.2, e:0.6	a:0.8, e:0.6, d:0.2

4.1. Illustrations to construct Global Compact PUF-Tree in Distributed environment

The transactional tree in distributed locations are named as Local Compact PUF-Trees (LPUF). Consider Table 3 as the Sample uncertain database of two different locations and PUF-Trees are constructed in distributed locations. $LPUF_1$, $LPUF_2$ are shown in Fig. 4 and Fig. 5 respectively.



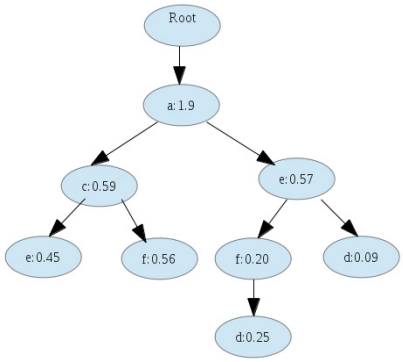


Fig. 4. PUF-Tree for Location 1 ($LPUF_1$) Fig.5. PUF-Tree for Location 2 (LPUF2)

4.2. Global Compact PUF-Tree

The proposed method for constructing GPUF-Tree contains two phases. The first phase is the distributed activity at each location based on trigger raised by the central server. The second phase is to Merge $LPUF_i$ it is shown in the following algorithm. This section focused on proposed model to obtain GPUF-Tree which further leads to derive multilevel information by deriving local and global knowledge for mining frequent patterns in a distributed environment.

Algorithm – Global PUF-Tree Construction.
Input: $I\text{-}List_i$, $LPUF_i$ where i=1 to n
Output: GPUF

1) Combine $I\text{-}List_1$, $I\text{-}List_2$,…….. $I\text{-}List_n$, as G-List
2) for i=1 to n
3) do
4) $LPUF_i$=Construct($PUF\text{-}Tree_i$,G-List)
5) GPUF=MergeLPUF($LPUF_1$, $LPUF_2$, $LPUF_3$,……. $LPUF_n$)
6) Done

Merge Tree(MergeLPUF)
Input: $LPUF_i$ where i=1 to n
Output: GPUF
Method:

(i) If n=2
    GPUF=Merge($LPUF_1$, $LPUF_2$)
(ii) Else
    a. $GPUF_1$=MergeLPUF($LPUF_1$, $LPUF_2$, $LPUF_3$,……. $LPUF_{n/2}$)
    b. $GPUF_2$= MergeLPUF($LPUF_{[n/2]+1}$,…….. $LPUF_n$)
    c. GPUF=Merge($GPUF_1$, $GPUF_2$)

Function Merge
Input: $LPUF_1$, $LPUF_2$
Output: GPUF

Method

1. Initialize GPUF= $LPUF_1$
2. for each branch br_i in $LPUF_2$
3. do
   a) find br_j in GPUF having same nodes at beginning with br_i
   b) if br_j exists then
      i. let $br_i=s.y_i$, $br_j=s.y_j$, where 's' is maximal prefix containing the common beginning nodes of br_i,br_j
      ii. if 's' is nonempty
      iii. then
      -Update I^{cap} of 's' in br_j path with I^{cap} of 's' in br_i path of $LPUF_2$
      -Attach y_i with its I^{cap} as a child node to last node of 's' in br_j
   c. else
      i. add br_i as new branch to the root node of GPUF with its I^{cap}
4. done

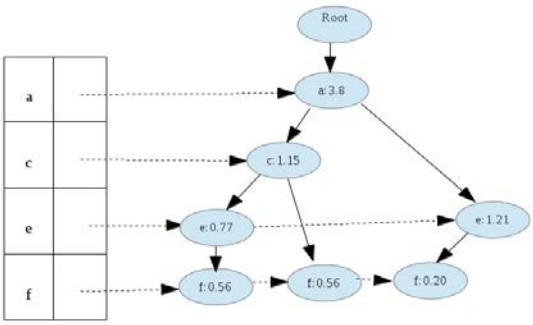


Fig. 1. GPUF-Tree

5. Experimentation and Analysis

To evaluate existing and proposed algorithms Compact PUF-Tree and GPUF Trees, experiments are conducted using Synthetic Data Set. They are executed in Intel processors on Java platform by Message Passing Interface and different libraries like JOMP and Xstream are used.

5.1. Experimentation on Construction of GPUF for various support thresholds

Considering 2000 transactions, data is divided into five parts. The data is distributed to five different nodes and sent to different processsors using threads. The construction of transactional trees $LPUF_1$ to $LPUF_5$ is carried out using OpenMP. The synchronization between processors is achieved internally using OpenMP. Table 4 shows the results of execution time to construct GPUF-Tree for various support thresholds. Fig. 7 implies that number of items will monotonically decrease as threshold increases. Table 4 and Fig. 7 are shown below at one location.

Table 1. Execution Time for constructing GPUF-Tree for various support thresholds

Min Sup	GPUF-Tree
5	122
10	47
15	42
20	36
25	34
30	29
35	19

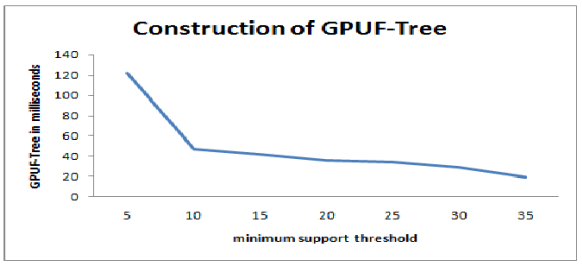


Fig. 2. Execution Time for constructing GPUF-Tree for various support thresholds

5.2. Construction of GPUF-Tree in Sequential and Parallel environment

The following is the sequential and parallel implementation of Compact PUF-Tree in order to achieve GPUF-Tree. The procedure (steps) which discussed earlier in Section 4 to construct GPUF-Trees in parallel environment is used to construct LPUF-Trees by considering five processors. These trees are merged in order to obtain GPUF-Tree.

Table 2. Computation time for construction GPUF-Tree for the given number of transactions with a support 4% in sequential and parallel implementation

Number of Transactions	Construction of PUF-Tree		Merging Time	Construction of GPUF-Tree	
	S	P		S	P
400	5.03	1.162	2.769	5.03	4.162
800	5.412	1.653	2.871	5.412	4.524
1000	6.155	0.667	4.567	6.155	5.234
2000	8.526	2.205	6.765	8.526	6.321

Table 5. shows the computation time to construct GPUF-Tree in Sequential (S) and Parallel (P) environments. Fig 8. shows the corresponding graph. The following are the observations:

- As the transactions are increasing, the elapsed time is also increasing.
- In a sequential mode, the time is increasing exponentially where as in parallel mode, the rate of increase is gradual.
- Expected reduction in elapsed time is not simply proportional to number of nodes, due to merging to be performed at central node.

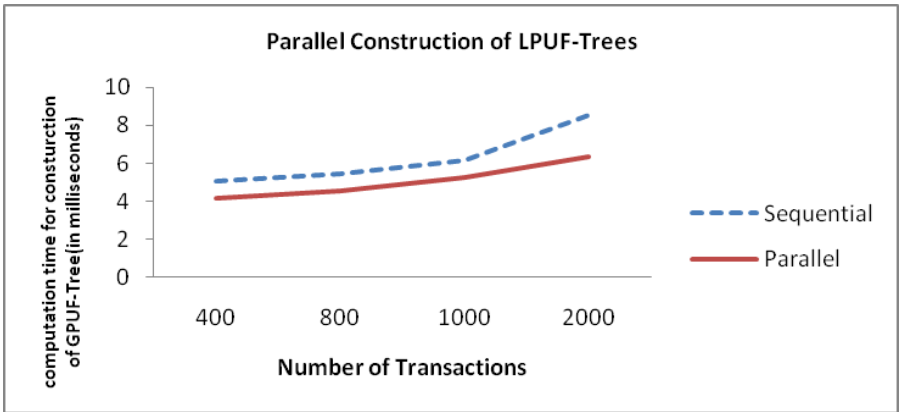


Fig. 3. Execution time for constructing GPUF-Tree by varying data size

6. Conclusions

This paper addressed some of the related issues which are centric to the transaction logs, mainly the uncertain data. This paper also studied on tree-based structure for representing the uncertain data and proposed the Distributed and Parallel construction of PUF-Tree. A model is proposed to achieve the performance gain during tree construction that can be effectively used in distributed environment. The proposed framework to obtain Global Compact PUF-Tree is given and the facts to obtain local and global frequent patterns from the abstraction of the local data are discussed. These patterns can be used in decision support systems. The present study discovers the global knowledge by privacy preserving in distributed environment.

References

1. Chui, C.-K., Kao, B., Hung, E.: Mining frequent itemsets from uncertain data. In: Zhou, Z.-H., Li, H., Yang, Q. (eds.) PAKDD 2007. LNCS (LNAI), vol. 4426, pp. 47–58. Springer, Heidelberg (2007)
2. Dai, X., Yiu, M.L., et al.: Probabilistic spatial queries on existentially uncertain data. In: Bauzer Medeiros, C., Egenhofer, M.J., Bertino, E. (eds.) SSTD 2005. LNCS, vol. 3633, pp. 400–417. Springer, Heidelberg (2005)
3. Leung, C.K.-S., Carmichael, C.L., Hao, B.: Efficient mining of frequent patterns from uncertain data. In: Proc. IEEE ICDM Workshops, pp. 489–494 (2007)
4. C. K.-S. Leung, M. A. F. Mateo, D. A. Brajczuk. A Tree-Based Approach for Frequent Pattern Mining from Uncertain Data, PAKDD 2008.
5. Chun Kit Chui and Ben Kao. A decremental approach for mining frequent itemsets from uncertain data. In The 12th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), pages 64-75, 2008.
6. C. C. Aggarwal, Y. Li, J. Wang, and J. Wang, "Frequent pattern mining with uncertain data," in *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. New York, NY, USA: ACM, 2009, pp. 29–38.
7. Leung, C.K.-S., Tanbeer, S.K.: Fast tree-based mining of frequent itemsets from uncertain data. In: Lee, S.-g., Peng, Z., Zhou, X., Moon, Y.-S., Unland, R., Yoo, J. (eds.) DASFAA 2012, Part I. LNCS, vol. 7238, pp. 272–287. Springer, Heidelberg (2012)
8. Leung CK, Tanbeer SK. PUF-tree: a compact tree structure for frequent pattern mining of uncertain data. In: Proceedings of the PAKDD 2013, Part I. Springer; 2013, p. 13–25.
9. Li Li, Donghai Zhai, Fan Jin, " A Parallel Algorithm for Frequent Itemset Mining", In Proc. IEEE International Conference on Parallel and Distributed Computing, Applications and Technologies, PDCAT 2003.
10. Han J., Pei J., Yin Y., "Mining Frequent Patterns without Candidate Generation". In: Proc. ACM Special Interest Group Management of Data (SIGMOD) International Conference, Dallas, 2000.